Symbolic Negotiation with Linear Logic

Peep Küngas[1] and Mihhail Matskin[2]

[1] Norwegian University of Science and Technology
Department of Computer and Information Science
Trondheim, Norway
peep@idi.ntnu.no
[2] Royal Institute of Technology
Department of Microelectronics and Information Technology
Kista, Sweden
misha@imit.kth.se

Abstract. Negotiation over resources and multi-agent planning are important issues in multi-agent systems research. It has been demonstrated [19] how symbolic negotiation and distributed planning together could be formalised as distributed Linear Logic (LL) theorem proving. LL has been chosen mainly because of its expressive power for representation of resources and its computation-oriented nature. This paper extends the previous work by taking advantage of a richer fragment of LL and introducing two sorts of nondeterministic choices into negotiation. This allows agents to reason and negotiate under certain degree of uncertainty. Additionally, a way of granting unbounded access to resources during negotiation is considered. Finally we extend our framework with first-order LL for expressing more complex offers during negotiation.

1 Introduction

Although the idea of distributed theorem proving as a formalism for agent negotiation is not new, there are still many open issues which require special attention. In particular, not much is known about limits of logics when it comes to capturing encodings of certain offers and reasoning about specific dialogues. Another important issue is computation—in order to ensure finiteness and efficiency of negotiations we may have to develop a special technique for more efficient proof search. This paper contributes to both issues—we extend expressiveness of previous work based on LL and introduce new LL inference figures, which allow us simplify proof search in our fragment of LL.

It was argued in [19] that distributed Linear Logic (LL) [10] theorem proving could be applied for symbolic agent negotiation. There was also proposed a formal mechanism for generating new offers. A corresponding framework allows agents to negotiate over resources and exploit capabilities of their partners. Since all participating agents have to achieve their personal goals, each agent has to be sure about resources that can be given away and capabilities that could be executed by other agents. Agent reasoning in [19] is an interactive process involving Partial Deduction (PD) and LL theorem proving. PD is used there

as a method of deducing subproblems, which from negotiation point of view are interpreted as offers. However, it was assumed that all offers, that an agent distributes, are independent from each-other.

In this paper we augment the previous approach with a mechanism for producing multiple interdependent offers at once. This provides competing agents with some hints about other distributed offers and available resources. Although the LL fragment in [19] permits usage of nondeterministic offers, it was not described how these offers are generated. We cover this aspect in our paper.

While in [19] the propositional intuitionistic multiplicative additive fragment of LL was considered, here we take advantage of full intuitionistic LL. This allows us to describe unbounded access to certain resources. Additionally we take advantage of first-order intuitionistic LL as a negotiation language.

Although several articles discuss language and representation issues of symbolic negotiation, we are more concerned with the computational side of a negotiation process. This paper presents a formalism for generating new offers using PD during negotiation. We define PD steps as inference figures in LL. While using those inference figures instead of basic LL rules, we can achieve more efficient proof search. These inference figures represent specialisation of LL to symbolic negotiation.

The paper is organized as follows. In Section 2 we present a general model of distributed problem solving and give an introduction to LL and PD. Additionally the LL formalisation to negotiation is given there. Section 3 describes a motivating example and illustrates how negotiation between agents works within our formalism. Section 4 presents new PD steps for agent negotiation. Section 5 reviews related work. The last section concludes the paper and discusses future work.

2 Formalisation of negotiation

2.1 Agent communication architecture

In this paper we consider self-interested cooperative agents. This means that agents cooperate with each other as long as it does not prevent them to achieve their own goals.

We define offers as the following structure:

$$(id_{req}, S, R, O),$$

where id_{req}, S, R and O denote message identifier, sender, receiver and offer respectively. Message identifier is needed to keep track of different negotiations. Sender and receiver are identifiers of participating agents and offer is represented with a LL formula.

While naming message identifiers, we use the following conventions:

- if agent $\mathcal{A}$ sends out an initial offer, then a is its message identifiers name
- if a includes an offer, then a' includes a counter-offer

– if an agent $\mathcal{A}$ sends out more than one offer, then their messages are indexed as $a_1, a_2, \ldots$

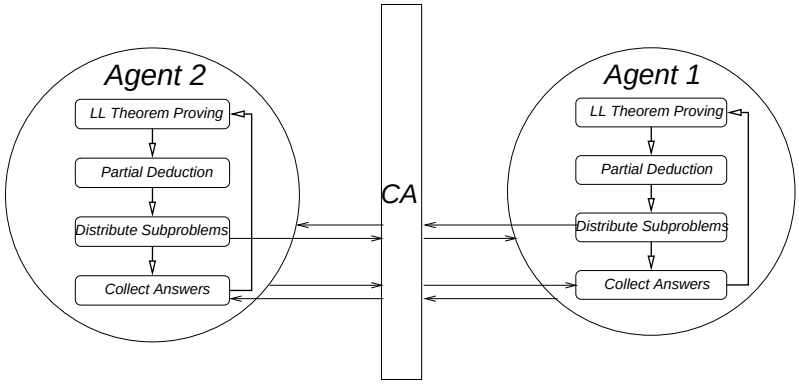

Fig. 1. General CPS model.

Our general problem solving model is presented in Figure 1. In this model each agent initially tries to solve its problem alone. If the agent cannot find a solution then subproblems are generated. The subproblems are distributed among the partners and they are treated as offers to other agents. In other words, they present what an agent can provide and what it expects to get in return. *CA* denotes to Communication Adapter, which translates offers from one agent such that a receiver could understand it. Formalisation of *CA* is given in Section 2.6.

2.2 Linear logic

LL is a refinement of classical logic introduced by J.-Y. Girard to provide means for keeping track of "resources". In LL two assumptions of a propositional constant A are distinguished from a single assumption of A. This does not apply in classical logic, since there the truth value of a fact does not depend on the number of copies of the fact. Indeed, LL is not about truth, it is about computation.

In the following we are considering intuitionistic fragment of LL (ILL) consisting of multiplicative conjunction ($\otimes$), additive disjunction ($\oplus$), additive conjunction ($\&$), linear implication ($\multimap$) and "of course" operator ($!$). In terms of resource acquisition the logical expression $A \otimes B \vdash C \otimes D$ means that resources C and D are obtainable only if both A and B are obtainable. After the sequent has been applied, A and B are consumed and C and D are produced.

The expression $A \vdash B \oplus C$ in contrast means that, if we have resource A, we can obtain either B or C, but we do not know which one of those. The expression $A \& B \vdash C$ on the other hand means that while having resources A and B we

can choose, which one of them to trade for C. Therefore it is said that $\oplus$ and & represent respectively *external* and *internal* choice.

In order to illustrate the above-mentioned features we can consider the following LL sequent from [21]—$(D \otimes D \otimes D \otimes D \otimes D) \vdash (H \otimes C \otimes (O \& S) \otimes !F \otimes (P \oplus I))$, which encodes a fixed price menu in a fast-food restaurant: for 5 dollars (D) you can get an hamburger (H), a coke (C), either onion soup O or salad S depending, which one *you* select, all the french fries (F) you can eat plus a pie (P) or an ice cream (I) depending on availability (restaurant owner selects for you). The formula $!F$ here means that we can use or generate a resource F as much as we want—the amount of the resource is unbounded.

To increase the expressiveness of formulae, we use the following abbreviation $a^n = \underbrace{a \otimes \ldots \otimes a}_{n}$, for $n > 0$.

2.3 Agents in LL

An agent is presented as the following LL sequent:

$$\Gamma; S \vdash G,$$

where Γ is a set of extralogical LL axioms representing agent's capabilities, S is the initial state and G is the goal state of the agent. Both S and G are multiplicative conjunctions of literals. Every element of Γ has the form

$$\vdash I \multimap O,$$

where I and O are formulae in conjunctive normal form which are, respectively, consumed and generated when a particular capability is applied. It has to be mentioned that a capability can be applied only, if conjuncts in I form a subset of conjuncts in S. It should be also underlined that in order to achieve their goals, agents have to construct (and then execute) the following program/plan from the elements of Γ:

$$\vdash S \multimap G.$$

2.4 Partial deduction and LL

Partial deduction (PD) (or partial evaluation of logic programs first introduced in [17]) is known as one of optimisation techniques in logic programming. Given a logic program, partial deduction derives a more specific program while preserving the meaning of the original program. Since the program is more specialised, it is usually more efficient than the original program, if executed. For instance, let A, B, C and D be propositional variables and $A \multimap B$, $B \multimap C$ and $C \multimap D$ computability statements in LL. Then possible partial deductions are $A \multimap C$, $B \multimap D$ and $A \multimap D$. It is easy to notice that the first corresponds to forward chaining (from initial states to goals), the second to backward chaining

(from goals to initial states) and the third could be either forward or backward chaining.

Although the original motivation behind PD was to deduce specialised logic programs with respect to a given goal, our motivation for PD is a bit different. We are applying PD for determining subtasks, which cannot be performed by a single agent, but still are possibly closer to a solution than an initial task. This means that given a state S and a goal G of an agent we compute a new state S' and a new goal G'. This information is forwarded to another agent for further inference. From PD point of view this means that the program $\Gamma \vdash S' \multimap G'$ would be derived from $\Gamma \vdash S \multimap G$. Then the derived program is sent to other entities, who modify it further.

The main problem with PD in LL is that although new derived states and goals are sound with respect to an initial specification, they may not preserve completeness anymore. This is due to resource-consciousness of LL—if a wrong proof branch is followed, initial beliefs may be consumed and thus further search becomes more limited. Therefore agents have to search, in the worst case, all possible PDs of initial specification to preserve completeness of distributed search mechanism. In [22] completeness and soundness issues of PD are considered for classical logic programs. Issues of complexity, completeness and soundness of PD in LL will be considered within another paper.

The following LL inference figures, $\mathcal{R}_b(L_i)$ and $\mathcal{R}_f(L_i)$, were defined in [19] for PD back- and forward chaining steps respectively:

$$\frac{S \vdash B \otimes C}{S \vdash A \otimes C}\ \mathcal{R}_b(L_i) \qquad \frac{A \otimes C \vdash G}{B \otimes C \vdash G}\ \mathcal{R}_f(L_i)$$

L_i in the inference figures is a labelling of a particular LL axiom representing an agent's capability (computability clause in PD) in the form $\vdash B \multimap_{L_i} A$. $\mathcal{R}_f(L_i)$ and $\mathcal{R}_b(L_i)$ apply clause L_i to move the initial state towards the goal state or the other way around. A, B and C are formulae in ILL.

In $\mathcal{R}_b(L_i)$ inference figure formulae $A \otimes C$ and $B \otimes C$ denote respectively goals G and G'. The inference figure encodes that, if there is an extralogical axiom $\vdash B \multimap A$, then we can change goal $A \otimes C$ to $B \otimes C$. Analogously, in the inference figure $\mathcal{R}_f(L_i)$ formulae $B \otimes C$ and $A \otimes C$ denote states S and S' respectively. The inference figure encodes that, if there is an extralogical axiom $\vdash B \multimap A$, then we can change initial state $B \otimes C$ to $A \otimes C$.

Although the defined PD steps consider only application of agent capabilities, they can be used for modelling resource exchange as well. We model exchange of resources with execution of capabilities, which generate those resources.

In addition to $\mathcal{R}_b(L_i)$ and $\mathcal{R}_f(L_i)$ we define other PD steps for constructing nondeterministic offers, to handle first-order representation and nondeterminism arising from usage of unbounded resources. Finally we introduce macros for more efficient PD.

2.5 Encoding offers in LL

Harland and Winikoff [12] presented the first ideas about applying LL theorem proving for agent negotiation. The main advantages of LL over classical logic is its resource-consciousness and existence of two kinds of nondeterminism. Both internal and external nondeterminism in negotiation rules can be represented. In the case of internal nondeterminism a choice is made by resource provider, whereas in the case of external nondeterminism a choice is made by resource consumer. For instance, formula $Dollar^5 \multimap Beer \oplus Soda$ (at the offer receiver side) means that an agent can provide either some *Beer* or *Soda* in return for 5 dollars, but the choice is made by the provider agent. The consumer agent has to be ready to obtain either a beer or a soda. The formula $Dollar \multimap Tobacco \& Lighter$ (again at the offer receiver side) in contrary means that the consumer may select which resource, *Tobacco* or *Lighter*, s/he gets for a *Dollar*.

In the context of negotiation, operators & and $\oplus$ have symmetrical meanings—what is $A \oplus B$ for one agent, is $A \& B$ to its partner. This means that if one agent gives an opportunity to another agent to choose between A and B, then the former agent has to be ready to provide both choices, A and B. When initial resources owned by agents and expected negotiation results have been specified, LL theorem proving is used for determining the negotiation process.

In [19] the ideas of Harland and Winikoff were augmented by allowing trading also services (agent capabilities). This is a step further toward the world where agents not only exchange resources, but also work for other agents in order to achieve their own goals. We write $A \vdash B \multimap C$ to indicate that an agent can trade resource A for a service $B \multimap C$. $B \multimap C$ denotes to a service, which consumes B and generates C.

There is another kind of nondeterministic construction in LL, namely the ! operator. Since $!A$ means that an agent can generate as many copies of A as required, the number of literals A is unbounded and represents additional kind of nondeterminism. From negotiation point of view, $!A$ represents unbounded access to the resource.

2.6 Communication adapter

In [24] bridge rules are used for translating formulae from one logic to another, when agents exchange offers. We adopt this idea of Communication Adapter (CA) for two reasons. First, it would allow us to encapsulate agents' internal states and, second, while offers are delivered by one agent to another, viewpoint to the offer is changing and internal and external choices are inversed. By viewpoint we mean an agent's role, which can be either receiver or sender of an offer.

The *CA* rule is described as follows. As long as formulae on the left and the right hand side of sequents consist of only $\otimes$ and $\multimap$ operators, the left and the right hand sides of sequents are inversed. However, if formulae contain disjunctions, their types have to be inversed as well. This has to be done because there are 2 disjunctions in LL—one with internal and another with external

choice. Since internal and external choices are context-dependent, they have to be inversed, when changing viewpoints. For instance, sequent $A \otimes (A \multimap B) \vdash C \oplus D$ is translated to $C \& D \vdash A \otimes (A \multimap B)$ by the CA rule:

$$\frac{\&_j B_j \vdash \bigoplus_i A_i}{\&_i A_i \vdash \bigoplus_j B_j}\ CA$$

In the CA rule A and B consist of multiplicative conjunctions and linear implications. We allow & only in the left hand side and $\oplus$ only in the right hand side of a sequent. Due to LL rules $R\&$ and $L\oplus$ the following conversions are allowed:

$$D \vdash \&_j D_j \Longrightarrow \bigcup_j (D \vdash D_j)$$

$$\bigoplus_j D_j \vdash D \Longrightarrow \bigcup_j (D_j \vdash D)$$

Therefore we do not lose in expressive power of LL, when limiting the syntax of offers in that way. Although this bridge rule is intended for agents reasoning in LL only, additional bridge rules may be constructed for communication with other non-LL agents.

3 A motivating example

For illustration of negotiation process we consider the following example. Let us have 3 agents representing a musician $\mathcal{M}$, a writer $\mathcal{W}$ and an artist $\mathcal{A}$. They all have personal goals they would like to achieve. We would like to emphasise that this example is supposed to demonstrate syntactical and computational aspects only and no pragmatic issues are considered here.

The musician would like to go out with her husband and therefore needs 2 concert tickets. Unfortunately the concert, she is interested in, has been sold out and therefore the only way to acquire the tickets is to ask them from other agents. In return she can grant a certain book and unlimited access to digital version of her albums. Thus

$$G_{\mathcal{M}} = \{Ticket^2\},$$

$$S_{\mathcal{M}} = \{!MP3 \otimes Book\}$$

and

$$\Gamma_{\mathcal{M}} = \emptyset.$$

The artist has promised to perform at a conference and thus needs 2 hours of background music and an MP3 player. Since the performance takes place at the same time as the concert he can give away the concert ticket. Formally,

$$G_{\mathcal{A}} = \{Perf\},$$

$$S_{\mathcal{A}} = \{Ticket\}$$

and

$$\Gamma_{\mathcal{A}} = \{\vdash MP3^2 \otimes MP3Player \multimap Perf\}.$$

The writer wants to relax and this can be achieved by reading a book and listening to music. He has both—a CD player and an MP3 player. Additionally he can write CDs from MP3 files. He also has a ticket to the same concert with the artist. However, he prefers staying at home this time. Thus formally this is described as follows:

$$G_{\mathcal{W}} = \{Relaxed\},$$

$$S_{\mathcal{W}} = \{Ticket \otimes CDPlayer \otimes MP3Player\},$$

$$\Gamma_{\mathcal{W}} = \begin{array}{l} \vdash (CD \otimes CDPlayer) \oplus (MP3 \otimes MP3Player) \multimap Music, \\ \vdash Music \otimes Book \multimap Relaxed, \\ \vdash MP3 \multimap CD. \end{array}$$

Our representation language so far differs from [19] by additional usage of $\oplus$ and !. While ! allows representing unbounded usage or access to a resource, $\oplus$ represents nondeterministic choice, which will be discussed below in more details.

Let us describe now the symbolic negotiation process between these 3 agents. The negotiation is initiated by agents $\mathcal{M}$ and $\mathcal{W}$. Agent $\mathcal{M}$ is unsure whether anyone has two tickets left to the concert. Therefore she decides to propose 2 separate offers instead of a single one and delivers them to $\mathcal{A}$ and $\mathcal{W}$:

$$(m_1, \mathcal{M}, \mathcal{A}, !MP3 \oplus Book \vdash Ticket)$$

and

$$(m_2, \mathcal{M}, \mathcal{W}, !MP3 \& Book \vdash Ticket)$$

The first offer means that $\mathcal{M}$ gives $\mathcal{A}$ an opportunity to choose between $!MP3$ and $Book$. The proposal to $\mathcal{W}$, however, means that $\mathcal{W}$ could get either $!MP3$ or $Book$, but the choice is made by $\mathcal{M}$. This is intuitive since $\mathcal{M}$ has no idea whether $\mathcal{A}$ would choose either $!MP3$ or $Book$.

The proposals describe internal and external choices in LL and are represented with operators & and $\oplus$ respectively. While $\oplus$ from the sender's point of

view gives choice to the receiver, & is the opposite—the sender makes the decision of which resource to deliver. It should be mentioned that messages, when received by agents, are translated using *CA* rule.

Agent $\mathcal{W}$ sends out the following offer:

$$(w_1, \mathcal{W}, \mathcal{A}, \mathit{MP3Player} \vdash \mathit{MP3} \otimes \mathit{Book})$$

which means that he can trade an MP3 player and a book for 1 hour of MP3 music. The following responses can be received:

$$(m_1', \mathcal{A}, \mathcal{M}, \mathit{Ticket} \vdash !\mathit{MP3})$$

and

$$(w_1', \mathcal{A}, \mathcal{W}, \mathit{MP3} \vdash \mathit{MP3Player}).$$

Based on the message from $\mathcal{A}$ to $\mathcal{W}$, the latter generates a new offer:

$$(m_2', \mathcal{W}, \mathcal{M}, \mathit{Ticket} \vdash \mathit{Book}).$$

The message from $\mathcal{W}$ to $\mathcal{M}$ means that $\mathcal{W}$ is not satisfied with the proposal and wants to be more precise. Namely, he is ready to trade his ticket for the book. Fortunately, $\mathcal{A}$ has chosen $!\mathit{MP3}$ and the proposal from $\mathcal{W}$ to $\mathcal{M}$ can be satisfied now. Additionally $\mathcal{W}$ accepts the proposal from $\mathcal{A}$ and everybody is now satisfied.

The presented scenario describes only how and which proposals are exchanged. The methodology for constructing these offers deserves a special attention and is clarified in the following sections. However, we only define new PD steps. Which steps and in which order are chosen during theorem proving, depends actually on a PD strategy and is not covered here. This would be covered in another paper together with other formal results.

4 Additional PD steps for agent negotiation

In this section we describe additional PD steps, which are needed for generating offers in ILL. These PD steps allow construction of nondeterministic offers and to handle unbounded access to resources. Additionally special PD steps for first-order ILL are introduced.

4.1 Generating nondeterministic offers

Nondeterministic offers can be generated basically in two ways. First, there may exist a particular capability having nondeterministic effects. Second, an agent uses some internal mechanism for composing such offers from scratch. Since in the first case nondeterministic offers are achieved via basic PD forward and backward steps, we consider here only the second case.

In order to describe how offers $!MP3 \oplus Book \vdash Ticket$ and $!MP3 \& Book \vdash Ticket$ were achieved from $!MP3 \otimes Book \vdash Ticket^2$ in Section 3 we present the following LL proof (M, T and B stand for *MP3*, *Ticket* and *Book*, respectively):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{!M \vdash !M}\,Id \quad \dfrac{\dfrac{}{B \vdash B}\,Id}{B \vdash !M \oplus B}\,R\oplus(b)}{!M, B \vdash !M \otimes (!M \oplus B)}\,R\otimes \quad \dfrac{\dfrac{}{B \vdash B}\,Id \quad \dfrac{\dfrac{}{!M \vdash !M}\,Id}{!M \vdash !M \oplus B}\,R\oplus(a)}{!M, B \vdash B \otimes (!M \oplus B)}\,R\otimes}{!M \otimes B \vdash !M \otimes (!M \oplus B)}\,L\otimes \quad \dfrac{\dfrac{}{!M\&B \vdash T} \quad \dfrac{}{!M \oplus B \vdash T}}{(!M\&B), (!M \oplus B) \vdash T \otimes T}\,R\otimes}{!M \otimes B \vdash (!M \otimes (!M \oplus B))\&(B \otimes (!M \oplus B))}\,R\&}{!M \otimes B \vdash (!M\&B) \otimes (!M \oplus B)}\,Rewrite \quad \dfrac{(!M\&B) \otimes (!M \oplus B) \vdash T \otimes T}{}\,L\otimes}{!M \otimes B \vdash T \otimes T}\,Cut$$

The proof can be generalised to a Partial Deduction (PD) step. This step generates multiple nondeterministic offers at once. This forward chaining PD step, called *Branch* in other sections of the paper, is defined as follows:

$$\dfrac{\dfrac{\vdots}{\bigotimes_{i=1}^{n} A_i \vdash (\bigwith_{i=1}^{k} A_i) \otimes \ldots \otimes (\bigoplus_{i=l}^{n} A_i)}\,Rewrite \quad \dfrac{\dfrac{}{\bigwith_{i=1}^{k} A_i \vdash B_1} \quad \ldots \otimes (\bigoplus_{i=l}^{n} A_i) \vdash \bigotimes_{i=2}^{m} B_i}{(\bigwith_{i=1}^{k} A_i), \ldots \otimes (\bigoplus_{i=l}^{n} A_i) \vdash \bigotimes_{i=1}^{m} B_i}\,R\otimes \quad \dfrac{\dfrac{}{\bigoplus_{i=l}^{n} A_i \vdash B_m}}{\vdots}}{(\bigwith_{i=1}^{k} A_i) \otimes \ldots \otimes (\bigoplus_{i=l}^{n} A_i) \vdash \bigotimes_{i=1}^{m} B_i}\,L\otimes$$

$$\dfrac{}{\bigotimes_{i=1}^{n} A_i \vdash \bigotimes_{i=1}^{m} B_i}\,Cut$$

where $1 \leq l, k \leq n, n > 1$. While the right branch of that inference figure generates multiple nondeterministic offers at once, the left branch ensures consistency of the offers. *Rewrite* refers that the right hand side of a sequent is transformed to disjunctive normal form with respect to & operator. From negotiation point of view this represents higher priority of offers, which include &, at the offer receiver side.

The number of nondeterministic branches cannot be larger than n, since we have only n resources. Additionally, the number of $\oplus$-offers (at sender side) is not greater than $n/2$. The latter derives from 2 assumptions: (1) for a choice at least 2 literals are needed and (2) $\oplus$-offers (from proposer point of view) must not overlap (otherwise it may happen that 2 agents choose the same resource and conflicts may occur). Generally, we assume that we can insert as many constants 1 as needed to enlarge m, since m has to be greater than or equal to the number of branches, which is limited by n.

4.2 First-order offers

So far we have employed only the propositional part of ILL. Let us consider now the same example from Section 3 but we modify it by replacing $G_{\mathcal{M}}$, $\Gamma_{\mathcal{M}}$, $S_{\mathcal{W}}$ and $S_{\mathcal{A}}$ with the following formulae:

$$G_{\mathcal{M}} = \{Concert(c)\}$$

$$\Gamma_{\mathcal{M}} = \{\vdash \forall x \forall y. \mathit{Ticket}(x, y) \otimes \mathit{Ticket}(x, y + 1) \multimap \mathit{Concert}(x)\}$$

$$S_{\mathcal{A}} = \{\mathit{Ticket}(c, 4)\}$$

$$S_{\mathcal{W}} = \{\mathit{Ticket}(c, 5) \otimes \mathit{CDPlayer} \otimes \mathit{MP3Player}\}$$

Here $\mathit{Ticket}(x, y)$ denotes a ticket to concert x and seat y (for simplicity we assume that all places have unique sequential number—this can be easily modified to a row and places system). It means that the musician goes to a concert only, if she has 2 tickets to the concert c and, moreover, only tickets for seats next to each-other are accepted.

In our formulae we allow only usage of the universal quantifier $\forall$. Its intended meaning is rather *for any* than *for every*. Although agents may send out first order offers, which have not been instantiated yet, their current state must be ground. To illustrate the construction of first-order offers, let us consider how agent $\mathcal{M}$ should proceed (again we indicate predefined literal symbols with their first letters):

$$\dfrac{\dfrac{\dfrac{}{\forall y.T(c,y) \otimes T(c,y+1) \vdash \forall y.T(c,y) \otimes T(c,y+1)}\,Id \quad \dfrac{}{C(c) \vdash C(c)}\,Id}{\forall y.T(c,y) \otimes T(c,y+1) \otimes (T(c,y) \otimes T(c,y+1) \multimap C(c)) \vdash C(c)}\,L\multimap}{\forall x \forall y.T(c,y) \otimes T(c,y+1) \otimes (T(x,y) \otimes T(x,y+1) \multimap C(x)) \vdash C(c)}\,L\forall$$

$$\vdots$$

$$\dfrac{\dfrac{\dfrac{}{!M \otimes B \vdash \forall y.T(c,y) \otimes T(c,y+1)} \quad \dfrac{}{\vdash \forall x \forall y.T(x,y) \otimes T(x,y+1) \multimap C(x)}\,Axiom}{!M \otimes B \vdash \forall x \forall y.T(c,y) \otimes T(c,y+1) \otimes (T(x,y) \otimes T(x,y+1) \multimap C(x))}\,T\otimes \quad \vdots\ L\forall}{!M \otimes B \vdash C(c)}\,Cut$$

Thus a new offer was generated:

$$!M \otimes B \vdash \forall y.T(c,y) \otimes T(c,y+1).$$

However, if the current state/goal pair of agent $\mathcal{M}$ is described with

$$\mathit{Ticket}(c, 4) \otimes \mathit{Ticket}(c, 5) \vdash A,$$

where A is an arbitrary goal, then the following inference could be applied and a new offer $C(c) \vdash A$ is generated:

$$\dfrac{\dfrac{\dfrac{}{T(c,4) \otimes T(c,5) \vdash T(c,4) \otimes T(c,5)}\,Id \quad \dfrac{}{C(c) \vdash A}}{T(c,4) \otimes T(c,5) \otimes (T(c,4) \otimes T(c,5) \multimap C(c)) \vdash A}\,L\multimap}{T(c,4) \otimes T(c,5) \otimes (\forall x \forall y.T(x,y) \otimes T(x,y+1) \multimap C(x)) \vdash A}\,L\forall$$

$$\vdots$$

$$\dfrac{\dfrac{\dfrac{}{T(c,4) \otimes T(c,5) \vdash T(c,4) \otimes T(c,5)}\,Id \quad \dfrac{}{\vdash \forall x \forall y.T(x,y) \otimes T(x,y+1) \multimap C(x)}\,Axiom}{T(c,4) \otimes T(c,5) \vdash T(c,4) \otimes T(c,5) \otimes (\forall x \forall y.T(x,y) \otimes T(x,y+1) \multimap C(x))}\,R\otimes \quad \vdots\ L\forall}{T(c,4) \otimes T(c,5) \vdash A}\,Cut$$

We can generalise these inferences to PD steps:

$$\frac{S \vdash A \otimes C}{S \vdash B \otimes C}\ \mathcal{R}_b(L_i(\underline{x})) \qquad \frac{A \otimes C \vdash G}{B \otimes C \vdash G}\ \mathcal{R}_f(L_i(\underline{x}))$$

where $L_i(\underline{x})$ is defined as $\vdash (\forall \underline{x} A' \multimap_{L_i(\underline{x})} B')$ for $\mathcal{R}_b(L_i(\underline{x}))$ and as $\vdash \forall \underline{x}(B' \multimap_{L_i(\underline{x})} A')$ for $\mathcal{R}_f(L_i(\underline{x}))$. A, B, C are LL formulae. Additionally we assume that $\underline{a} \stackrel{def}{=} a_1, a_2, \ldots$ is an ordered set of constants, $\underline{x} \stackrel{def}{=} x_1, x_2, \ldots$ is an ordered set of variables, $[\underline{a}/\underline{x}]$ denotes substitution, and $X = X'[\underline{a}/\underline{x}]$. When substitution is applied, elements in $\underline{a}$ and $\underline{x}$ are mapped to each other in the order they appear in the ordered sets. These sets must have the same number of elements.

LL inference figures for $\mathcal{R}_b(L_i(\underline{x}))$ and $\mathcal{R}_f(L_i(\underline{x}))$ look as follows. Backward chaining step $\mathcal{R}_b(L_i(\underline{x}))$:

$$\dfrac{\dfrac{\overline{S \vdash A \otimes C}\ Axiom \quad \overline{\vdash (\forall \underline{x} A' \multimap_{L_i(\underline{x})} B')}}{S \vdash A \otimes C \otimes (\forall \underline{x} A' \multimap_{L_i(\underline{x})} B')}\ R\otimes \quad \dfrac{\dfrac{\dfrac{\overline{C \vdash C}\ Id \quad \dfrac{\dfrac{\overline{A \vdash A}\ Id \quad \overline{B \vdash B}\ Id}{A, (A \multimap_{L_i(\underline{a})} B) \vdash B}\ L\multimap}{A \otimes (A \multimap_{L_i(\underline{a})} B) \vdash B}\ L\otimes}{C, A \otimes (A \multimap_{L_i(\underline{a})} B) \vdash B \otimes C}\ R\otimes}{A \otimes C \otimes (A \multimap_{L_i(\underline{a})} B) \vdash B \otimes C}\ L\otimes}{A \otimes C \otimes (\forall \underline{x} A' \multimap_{L_i(\underline{x})} B') \vdash B \otimes C}\ L\forall}{S \vdash B \otimes C}\ Cut$$

Forward chaining step $\mathcal{R}_f(L_i(\underline{x}))$:

$$\dfrac{\dfrac{\dfrac{\overline{B \otimes C \vdash B \otimes C}\ Id \quad \overline{\vdash \forall \underline{x}(B' \multimap_{L_i(\underline{x})} A')}\ Axiom}{B \otimes C \vdash B \otimes C \otimes (\forall \underline{x} B' \multimap_{L_i(\underline{x})} A')}\ R\otimes \quad \dfrac{\dfrac{\dfrac{\dfrac{\overline{C \vdash C}\ Id \quad \dfrac{\dfrac{\overline{B \vdash B}\ Id \quad \overline{A \vdash A}\ Id}{B, (B \multimap_{L_i(\underline{a})} A) \vdash A}\ L\multimap}{B \otimes (B \multimap_{L_i(\underline{a})} A) \vdash A}\ L\otimes}{C, B \otimes (B \multimap_{L_i(\underline{a})} A) \vdash A \otimes C}\ R\otimes}{B \otimes C \otimes (B \multimap_{L_i(\underline{a})} A) \vdash A \otimes C}\ L\otimes}{B \otimes C \otimes (\forall \underline{x} B' \multimap_{L_i(\underline{x})} A') \vdash A \otimes C}\ L\forall \quad \overline{A \otimes C \vdash G}}{B \otimes C \otimes (\forall \underline{x} B' \multimap_{L_i(\underline{x})} A') \vdash G}\ Cut}{B \otimes C \vdash G}\ Cut$$

4.3 Unbounded access to resources

In order to manage access to unbounded resources, we need PD steps $\mathcal{R}_{C_l}$, $\mathcal{R}_{C_r}$, $\mathcal{R}_{L_l}$, $\mathcal{R}_{L_r}$, $\mathcal{R}_{W_l}$, $\mathcal{R}_{W_r}$. They are formalised as the following LL inference figures:

$$\frac{!A \otimes !A \otimes B \vdash C}{!A \otimes B \vdash C}\ \mathcal{R}_{C_l} \qquad \frac{C \vdash !A \otimes !A \otimes B}{C \vdash !A \otimes B}\ \mathcal{R}_{C_r}$$

$$\frac{A \otimes B \vdash C}{!A \otimes B \vdash C}\ \mathcal{R}_{L_l} \qquad \frac{C \vdash A \otimes B}{C \vdash !A \otimes B}\ \mathcal{R}_{L_r}$$

$$\frac{B \vdash C}{!A \otimes B \vdash C}\ \mathcal{R}_{W_l} \qquad \frac{C \vdash B}{C \vdash !A \otimes B}\ \mathcal{R}_{W_r}$$

Due to the lack of space we do not give proofs for these inference figures. However, they directly reflect LL rules $!C$, $!L$ and $!W$. Additionally we define 2 macros, $\mathcal{R}_{!_l}(n)$ and $\mathcal{R}_{!_r}(n)$, respectively, using the previously specified inference figures:

$$\dfrac{\dfrac{\dfrac{!A \otimes A^n \otimes B \vdash C}{\vdots}}{!A \otimes A \otimes B \vdash C}\,\mathcal{R}_{L_l}}{\dfrac{!A \otimes !A \otimes B \vdash C}{!A \otimes B \vdash C}\,\mathcal{R}_{C_l}} \qquad \dfrac{\dfrac{\dfrac{C \vdash !A \otimes A^n \otimes B}{\vdots}}{C \vdash !A \otimes A \otimes B}\,\mathcal{R}_{L_r}}{\dfrac{C \vdash !A \otimes !A \otimes B}{C \vdash !A \otimes B}\,\mathcal{R}_{C_r}}$$

5 Related work

As it has been indicated in [15] negotiation is the most fundamental and powerful mechanism for managing inter-agent dependencies at run-time. Negotiation may be required both for self-interested and cooperative agents. It allows to reach a mutually acceptable agreement on some matter by a group of agents.

Kraus et al [18] give a logical description for negotiation via argumentation for BDI agents. They classify arguments as threats and promises, which are identified as most common arguments in human negotiations. In our case only promises are considered, since in order to figure out possible threats to goals of particular agents, agents' beliefs, goals and capabilities should be known in advance to the persuader. We assume, that our agents do not explicitly communicate about their internal state.

Fisher [7] introduced the idea of distributed theorem proving in classical logic as agent negotiation. In his approach all agents share the common view to the world and if a new clause is inferred, all agents would sense it. Inferred clauses are distributed among agents via broadcasting. Then, considering the received information, agents infer new clauses and broadcast them further again. Although agents have a common knowledge about inferred clauses, they may hold different sets of inference rules. Distribution of a collection of rules between agents means that different agents may have different capabilities and make different inferences. The latter implies that different agents contribute to different phases of proof search. Our approach differs from that work mainly in 2 aspects (in addition to usage of another logic): (1) our agents do not share a common view of a world and (2) inference results are not broadcasted.

Parsons et al [24] defined negotiation as interleaved formal reasoning and arguing in classical logic. Arguments and contra arguments are derived using theorem proving whilst taking into consideration agents' own goals. Sadri et al [25] propose an abductive logic programming approach to automated negotiation, which is built on Amgoud et al [1] work on argumentation. The work of Sadri et al is more specialised and detailed than the work by Amgoud et al. That allows deeper analysis of the reasoning mechanism and the knowledge required to build negotiation dialogues.

There are some similarities between abduction and PD. However, while abduction is about finding a hypothesis to explain given results, then PD achieves the hypothesis as a side-effect. The latter could be explained by stating that in our case the given results are a part of a program and PD is about program transformation, not about finding an hypothesis. By taking into account the preceding, abduction could be implemented through PD.

Our approach could be viewed as distributed planning similarly to the work in [6]. Case-based planning has been used for coordinating agent teams in [9]. The planner generates, so called, a shared mental model of the team plan. Then all agents adapt their plans to the team plan. This work is influenced by the joint intentions [20, 4] and shared plans [11] theory.

In [3] LL has been used for prototyping multi-agent systems at conceptual level. Because of the fixed semantics of LL, it is possible to verify whether a system functions as intended at conceptual level. Although the prototype LL program is executable, it is still too high level to produce a final agent-based software. Thus another logic programming language is embedded to compose the final software.

Harland and Winikoff [13] address the question of how to integrate both proactive and reactive properties of agents into LL programming framework. They use forward chaining to model the reactive behaviour of an agent and backward chaining to model the proactive behaviour. This type of computation is called as mixed mode computation, because of both forward and backward chaining are allowed.

According to [2] our theorem proving methodology is characterised with *parallelism at the search level*. The approach relates by theorem proving methodology mostly to the successors of Team-Work [5, 8]. Fuchs [8] describes an approach, where distribution of facts and sub-problems is organised through request—that is the basic mechanism behind our methodology as well. However, the previous paper considers first-order logic with equality, which is somehow different from LL.

6 Conclusions and future work

In this paper we augmented a symbolic negotiation framework, which was initially introduced in [19]. More specifically, we introduced PD steps for generating nondeterministic offers and handling unbounded access to resources. We also extended the framework with first-order PD steps and thus increased expressiveness of offers. We defined PD steps as special LL inference figures. While applying these inference figures during proof search instead of basic LL rules we can gain higher efficiency.

We have implemented a planner on top of a first-order MILL theorem prover. Instead of using individual LL inference rules, the theorem prover applies the inference figures presented in this paper. Although that approach makes the prover application specific, it allows higher computational efficiency. The planner

is available at `http://www.idi.ntnu.no/~peep/RAPS`. In future we would like to extend the prover to cover ILL as well.

There are many open issues related to agent systems, negotiation and LL. Since LL has been extended with temporal properties [14, 16], we would like to introduce the notion of time to our framework as well. Additionally it has been indicated in [23] that the modal logic S4 has a direct translation to LL. This result is motivating for considering whether current BDI-theories could be embedded into our framework. We also have plans to work on application of our approach to some more practical cases of negotiation, in particular, related to web services selection and composition.

Acknowledgements

This work is partially supported by the Norwegian Research Foundation in the framework of the Information and Communication Technology (IKT-2010) program—the ADIS project. We would like to thank the anonymous referees for their comments.

References

1. L. Amgoud, S. Parsons, N. Maudet. Arguments, Dialogue and Negotiation. In Proceedings of 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20–25, 2000, pp. 338–342, IOS Press, 2000.
2. M. P. Bonacina. A Taxonomy of Parallel Strategies for Deduction. Annals of Mathematics and Artificial Intelligence, Vol. 29, No. 1–4, pp. 223–257, 2000.
3. M. Bozzano, G. Delzanno, M. Martelli, V. Mascardi, F. Zini. Logic Programming & Multi-Agent Systems: a Synergic Combination for Applications and Semantics. In The Logic Programming Paradigm: a 25-Year Perspective, pp. 5–32, Springer-Verlag, 1999.
4. P. R. Cohen, H. J. Levesque. Teamwork. Nous, Vol. 25, No. 4, pp. 487–512, 1991.
5. J. Denzinger, D. Fuchs. Cooperation of Heterogeneous Provers. In Proceedings of IJCAI-99, pp. 10–15, Morgan Kaufmann, 1999.
6. M. Fisher, M. Wooldridge. Distributed Problem-Solving as Concurrent Theorem Proving. In Proceedings of 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Ronneby, Sweden, May 13-16, 1997. Lecture Notes in Computer Science, Vol. 1237, pp. 128–140, Springer-Verlag, 1997.
7. M. Fisher. Characterising Simple Negotiation as Distributed Agent-Based Theorem-Proving—A Preliminary Report. In Proceedings of the Fourth International Conference on Multi-Agent Systems, Boston, July 2000, IEEE Press.
8. D. Fuchs. Requirement-Based Cooperative Theorem Proving. In Proceedings of JELIA-1998, Dagstuhl, Germany, October 12–15, 1998, Lecture Notes in Artificial Intelligence, Vol. 1489, pp. 139–153, Springer, 1998.
9. J. A. Giampapa, K. Sycara. Conversational Case-Based Planning for Agent Team Coordination. In D. W. Aha, I. Watson (eds). Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR 2001, July 2001, Lecture Notes in Artificial Intelligence, Vol. 2080, pp. 189–203, Springer-Verlag, 2001.

10. J.-Y. Girard. Linear Logic. Theoretical Computer Science, Vol. 50, pp. 1–102, 1987.
11. B. Grosz, S. Kraus. Collaborative Plans for Complex Group Actions. Artificial Intelligence, Vol. 86, pp. 269–357, 1996.
12. J. Harland, M. Winikoff. Agent Negotiation as Proof Search in Linear Logic. In Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), July 15–19, 2002, Bologna, Italy.
13. J. Harland, M. Winikoff. Language Design Issues for Agents based on Linear Logic. In Proceedings of the Workshop on Computational Logic in Multi-Agent Systems (CLIMA'02), August 2002.
14. T. Hirai. Propositional Temporal Linear Logic and its Application to Concurrent Systems. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Special Section on Concurrent Systems Technology), Vol. E83-A, No. 11, pp. 2219–2227, November 2000.
15. N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge. Automated Negotiation: Prospects, Methods and Challenges, International Journal of Group Decision and Negotiation, Vol. 10, No. 2, pp. 199–215, 2001.
16. M. Kanovich, T. Ito. Temporal Linear Logic Specifications for Concurrent Processes (Extended Abstract). In Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, 29 June–2 July 1997, pp. 48–57, IEEE Computer Society Press, 1997.
17. J. Komorowski. A Specification of An Abstract Prolog Machine and Its Application to Partial Evaluation. PhD thesis, Technical Report LSST 69, Department of Computer and Information Science, Linkoping University, Linkoping, Sweden, 1981.
18. S. Kraus, K. Sycara, A. Evenchik. Reaching Agreements through Argumentation: A Logical Model and Implementation. Artificial Intelligence, Vol. 104, No. 1–2, pp. 1–69, 1998.
19. P. Küngas, M. Matskin. Linear Logic, Partial Deduction and Cooperative Problem Solving. To appear in Proceedings of the First International Workshop on Declarative Agent Languages and Technologies, DALT 2003, Melbourne, Australia, July 15, Lecture Notes in Computer Science Series, Springer.
20. H. J. Levesque, P. R. Cohen, J. H. T. Nunes. On Acting Together. In Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI-90, pp. 94–99, 1990.
21. P. Lincoln. Linear Logic. ACM SIGACT Notices, Vol. 23, No. 2, pp. 29–37, Spring 1992.
22. J. W. Lloyd, J. C. Shepherdson. Partial Evaluation in Logic Programming. Journal of Logic Programming, Vol. 11, pp. 217–242, 1991.
23. S. Martini, A. Masini. A Modal View of Linear Logic. Journal of Symbolic Logic, Vol. 59, No. 3, pp. 888–899, September 1994.
24. S. Parsons, C. Sierra, N. Jennings. Agents that Reason and Negotiate by Arguing. Journal of Logic and Computation, Vol. 8, No. 3, pp. 261–292, 1998.
25. F. Sadri, F. Toni, P. Torroni. Logic Agents, Dialogues and Negotiation: An Abductive Approach. In Proceedings of the Symposium on Information Agents for E-Commerce, Artificial Intelligence and the Simulation of Behaviour Convention (AISB-2001), York, UK, March 21–24, 2001.